国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第16卷第1期 1994年3月 Vol. 16 No. 1

大型稀疏线性代数迭代库在YH机上的高效实现*

何新芳 胡庆丰 王丽萍 田泽荣

(国防科技大学电子计算机系 长沙 410073)

摘 要 本文讨论了大型稀疏线性代数方程组的迭代算法、加速方法、存贮技术及并行算法。结合向量机特点，采取有效程序优化措施，开发研制了标量和向量库程序。在YH系列机上试算结果表明：大型稀疏线性代数向量迭代库比标量迭代库速度有较大提高。当N≥100时，在YH－1机上加速比约2～8；在YH－2机上约2～7；当迭代次数增加时，加速比提高更明显；库中共轭梯度（CG）加速方法能有效地加快收敛，可减少迭代次数一半以上。

关键词 加速比，向量，标量

分类号 TP311，O241.6

在许多科学与工程计算中，经常遇到求解大型稀疏线性代数方程组的问题。求解此类问题的方法有直接求解和迭代求解两种。由于问题规模一般比较大，常采取压缩存贮格式。当系数矩阵为带状或块三对角等特殊类型时，直接解法和迭代解法都可较容易地实现并行计算，但当系数矩阵任意稀疏时，对其压缩存贮格式实现并行求解比较困难。研究系数矩阵任意稀疏时压缩存贮格式的并行算法，并研制出相应的向量或并行计算的应用软件，对充分发挥并行机特点，提高此类问题的计算效率，有着重要的实际意义和应用价值。

本文介绍了在并行算法研究的基础上，如何对系数矩阵为任意稀疏时，其压缩存贮格式的标量线性代数计算程序进行算法改造和程序优化，开发研制出向量大型稀疏迭代库。

1 基本内容

大型稀疏线性代数方程组迭代库主要是针对系数矩阵为一般稀疏和对称稀疏时，对角线元素轻度占优的线性代数方程组，采用Jacobi算法，逐次超松弛法（SOR），对称逐次超松弛法（SSOR），RS算法等迭代算法求解。为加速收敛，对每种算法又采取了不同的加速方法，如自适应切比雪夫（SI）加速法和自适应共轭梯度（CG）加速法；以及系数矩阵为块三对角或带状矩阵时，采用预处理共轭梯度法求解。整个库包括JCG（Jacobi共轭梯度）法、JSI（Jacobi半迭代）法、SOR（逐次超松弛）法、SSORCG（对称SOR

* 1993年5月5日收稿

共轭梯度）法、SSORSI（对称 SOR 半迭代）法、RSCG（约减系统共轭梯度）法、RSSI（约减系统半迭代）法、INCG（预处理共轭梯度）法等。

2 压缩存贮技术

由于在实践中遇到的稀疏线性代数方程组规模很大，而其系数矩阵元素大部分是零，非零元素一般只占 5%～20%。若将它们全部存贮，需要占用大量空间，给求解带来很大困难。本库算法均采用按行列指针压缩存贮技术。对原二维系数矩阵 A 的非零元素用三个一维数组存贮，一个一维实数组 AP 用于按行存贮 A 的非零元素，一个整数组 JA 存贮对应于 AP 中非零元素的列标，另一个整数组 IA 用于存贮行指针，存贮每一行的第一个非零元素在一维数组 AP 中的位置。程序中还设置了一些工作数组，用于实现快速元素存取和并行计算。采用压缩存贮格式经常遇到间接存取问题，给并行计算带来不便。

3 并行算法研究及程序优化

要把传统串行计算的稀疏线性代数方程组迭代库改造成并行计算的向量库，首先要研究并行算法。我们对原标量库中的串行计算程序进行系统的静态分析和大量的动态测试，找出影响运算速度的核心部分，再针对核心部分研究其并行算法及有效的实现措施。主要有以下几个方面：

（1）在迭代计算中经常遇到类似于稀疏矩阵乘、稀疏矩阵与向量乘、求稀疏向量点积之类的基本核心模块或程序段。这类模块在迭代库中有十多个，有的模块在多种算法中被反复调用。这些模块计算量大，费时多，有的在一种迭代算法中竟占计算量的 80% 以上，因此研究此类模块的并行计算及实现是提高迭代库运算速度的关键。例如稀疏矩阵与向量乘模块 PMULT 被七种算法调用，程序如下：

```
      SUBROUTINE PMULT(NN, IA, JA, AP, U, W)
      INTEGER IA(*), JA(*)
      REAL AP(*), U(NN), W(NN)
      非对称稀疏矩阵与向量乘
      DO 30 II=1, N
      IBGN=IA(II)
      IEND=IA(II+1)-1
      SUM=0.
      DO 10 JJ=IBGN,IEND
      JAJJ=JA(JJ)
10    SUM=SUM+AP(JJ)*U(JAJJ)
30    W(II)=SUM
      对称稀疏矩阵与向量乘
      DO 40 II=1, N
40    W(II)=0.
      DO 70 II=1, N
```

```
      IBGN=IA(II)
      IEND=IA(II+1)-1
      UII=U(II)
      WII=W(II)
      DO 50 JJ=IBGN,IEND
      JAJJ=JA(JJ)
      W(II)=WII+AP(JJ)*U(JAJJ)
50    W(JAJJ)=W(JAJJ)+AP(JJ)*UII
70    CONTINUE
      ……
      END
```

上述模块循环体中遇到了间接存取问题，计算相关性强，无法直接实现并行计算，效率很低。为了实现向量计算，我们将上述模块改造如下：

```
      非对称稀疏矩阵与向量乘：
      DO 30 II=1, N
      IBGN=IA(II)
      IEND=IA(II+1)-1
      IJ=IEND-IBGN+1
      CALL GATHER(IJ, AA(1), U, JA(IBGN))
30    W(II)=SDOT(IJ, AA(1), 1, AP(IBGN), 1)
      对称稀疏矩阵与向量乘：
      DO 70 II=1, N
      IBGN=IA(II)
      IEND=IA(II+1)-1
      UII=U(II)
      IJ=IEND-IBGN+1
      CALL GATHER(IJ, AA(1), U, JA(IBGN), 1)
      W(II)=SDOT(IJ, AA(1), 1, A(IBGN), 1)
      CALL GATHER(IJ, AA(1), W, JA(IBGN))
      DO 60 JJ=IBGN, IEND
60    AA(JJ-IBGN+1)=AA(JJ-IBGN+1)+A(JJ)*U(II)
      CALL SCATTE(IJ, W, JA(IBGN), AA(1))
70    W(II)=WII
      ……
      END
```

上述程序段中 AA(*)为增设的临时工作数组，GATHER，SCATTE，SDOT 为新研制的向量汇编模块（参见下[2]）。改造后的模块完全实现了并行计算，当 N=500，M=250（N 为稀疏矩阵阶，M 为稀疏系数矩阵上半带宽，下同）时，平均每次调用时间可减少到

原来的十一分之一以下（见表1），表1中还列出了另外七个类似模块的运行加速比。

（2）改造串行算法，研制高效向量汇编功能模块。

上文已指出，稀疏矩阵压缩存贮后带来了大量的间接存取问题，给并行计算带来诸多不便。为实现并行计算，首先必须研究并行间接存取技术，并研制出并行间接存取模块。为此，我们根据迭代库中遇到的各种间接存取问题，研制了高效向量间接存取子程序GATHER 和 SCATTE：

GATHER（N，A，B，INDEX），功能是实现向量间接取，相当于 A(I)＝B(INDEX(IJ))，I＝1，2，…，N。

SCATTE(N，B，INDEX，A)，功能是实现向量间接存，相当于 B(INDEX(I))＝A(I)，I＝1，2，…，N。

其次是对程序段中多次遇到的类似于求点积的程序段，如上述循环体中的 SUM＝SUM＋AP(JJ)＊U(JAJJ)，就是求点积。对此，我们结合并行机特点，充分利用向量功能部件，链接技术，循环特性及硬指令等，便用分段法分别研制出在 YH－1及 YH－2高效运行的向量汇编模块 SDOT。可调用 SDOT 代替类似于求点积的程序段。这样处理的结果有效地提高了整库的运行效率。

表1中给出了稀疏矩阵运算的标量和向量模块在 YH－1机上运行时间及加速比，在 YH－2机上类同。表2，t_s、t_v 单位为秒，加速比 $s_p=\frac{t_s}{t_v}$，符号意义下同。

（3）程序优化

我们采取的优化措施有以下几点：

a．凡是能用向量语句描述的循环体和表达式，一律用强行向量化的标量语句描述。

b．子程序插入，对迭代库中主模块和子模块经常调用的一些功能模块，如数组或向量运算，向量赋值，交换等功能模块，一律改造成强行向量化的标量语句，直接插入到调用该模块的程序段中，实现向量运算。

c．进行循环体分割，从而达到实现循环体中条件语句向量化。如对程序段：

```
      DO 50 I=IBGN, IEND
      JAI=JA(I)
      IF(JAI.GT.K) GOTO 40
      TEMP2=TEMP2-AP(I)*W1(JAI)
      GOTO 50
40    TEMP1=TEMP1-AP(I)*W1(JAI)
50    CONTINUE
```

可增设临时工作数组 AA(＊)，调用间接取模块，改造为：

```
      CALL GATHER(IJ, AA(1), W1(1), JA(IBGN))
      DO 50 I=IBGN, IEND
      AA(I-IBGN+1)=-AP(I)*AA(I-IBGN+1)
      TEMP1=TEMP1+CVMGT(AA(I-IBGN+1), 0, JA(I), LE, K)
50    TEMP2=TEMP2+CVMGT(AA(I-IBGN+1), 0, JA(I), LE, K)
```

4 效能和算法分析

为了确保大型稀疏线性代数并行迭代库正确可靠运行，我们在开发研制向量库的同时还研制出了相应的测试库，用于测试各种算法和程序运行的正确性以及各种情况下的运行效率等。利用测试库进行了大量的数据试验，测试结果表明：向量库与标量库各种算法的迭代次数与计算精度完全一致，但向量库运算速度快，效率高。有些算法向量库与标量库加速比高达7～8。表2中列出了 N＝700，上半带宽 M＝250时，各种算法在 YH－1和 YH－2机上标量与向量的运行时间、迭代次数及加速比。从表2中可看出，随迭代次数增加，加速比增大。共轭梯度加速法较切比雪夫半迭代加速法能更有效地加速收敛，通常情况下，共轭梯度加速法可减少迭代次数一半以上。表3中列出了向量库和标量库在 YH－1机及 YH－2机上整库运行情况。表3中数据说明向量库比标量库运行速度快，效率高。当 N＝500，600，…，1000，而 M＝1，100，300时，整库（包含十四种算法，三种带宽）运行时间总和的加速比，在 YH－1机上平均4.5，在 YH－2机平均为5.1。

表 1　稀疏矩阵标量和向量运算模块在 YH－1机上运行加速比

模块名	N＝500　M＝250			N＝800　M＝400		
	标量 t_s	向量 t_v	加速比 s_p	标量 t_s	向量 t_v	加速比 s_p
PBBETA	1.615665	0.137923	11.71	4.127598	0.315211	13.09
PSSORI	1.559739	0.145936	10.69	3.970264	0.299818	13.24
PFSOR	0.7792339	0.072612	10.73	1.983419	0.149285	13.29
PVTBV	0.777925	0.070397	11.05	1.98324	0.145748	13.61
PMULT	0.778517	0.07048	11.06	1.987414	0.145625	13.65
UNSCAL	1.148798	0.094161	12.2	2.92409	0.217392	13.45
PJAC	0.777017	0.071966	10.8	1.98202	0.148433	13.35
PRSBLK	0.260512	0.029764	8.75	0.66327	0.061513	10.78

表 2　N＝700，M＝250时向量和标量库运行情况比较

模块名	迭代次数	标量库运行时间(秒)		向量库运行时间(秒)		加速比	
		YH－1(T_{1s})	YH－2(T_{2s})	YH－1(T_{1v})	YH－2(T_{1v})	YH－1 T_{1s}/T_{1v}	YH－2 T_{2s}/T_{2v}
对称 JSI	49	37.83	13.59	10.94	3.271	3.46	4.15
非对称 JSI	49	69.65	22.23	9.339	2.712	7.46	8.20
对称 JCG	13	12.84	4.421	4.167	1.157	3.07	3.82
非对称 JCG	18	30.40	9.434	5.694	1.473	5.34	6.40
对称 SOR	38	49.44	15.88	10.58	3.055	4.67	5.20
非对称 SOR	38	54.12	17.41	7.921	2.223	6.83	7.82
对称 SSORSI	20	52.35	17.11	9.146	2.735	5.72	6.26
非对称 SSORSI	20	87.95	26.14	14.13	4.128	6.22	6.33
对称 SSORCG	10	36.86	11.72	8.398	2.373	4.39	4.94
非对称 SSORCG	10	50.74	14.93	10.68	2.834	4.75	5.27
对称 RSSI	21	28.25	9.089	6.071	2.017	4.65	4.51
非对称 RSSI	22	33.47	10.52	7.865	2.114	4.26	4.98
对称 RSCG	5	11.57	3.485	4.218	1.070	2.74	3.26
非对称 RSCG	5	15.23	4.401	5.705	1.314	2.67	3.35

表 3　向量库和标量库整库运行情况比较

N	标量运行时间(秒)		向量库运行时间(秒)		加速比	
	YH−1(T_{1s})	YH−2(T_{2s})	YH−1(T_{1v})	YH−2(T_{2v})	T_{1s}/T_{1v}	T_{2s}/T_{2s}
500	554.527	175.846	129.315	36.352	4.29	4.84
600	730.347	231.046	164.255	45.901	4.45	5.03
700	879.684	278.272	195.885	54.562	4.49	5.10
800	1033.62	327.995	228.55	63.626	4.52	5.16
900	1216.66	385.684	267.19	73.53	4.55	5.25
1000	1399.32	445.906	303.57	84.723	4.61	5.26

参 考 文 献

1 Young D M, Jea K C. Conjugate Gradient Acceleration of Iterative Methods: Part II. The Nonsymmetrizable Case Rep CNA−163, Center for Numerical Analysis, Univ. of Texas at Austin, 1980

2 Young D M, Hayes L, Jea K C. Conjugate Gradient Acceleration of Iterative Methods: Part I. The Symmetrizable Case Rep CNA−162, Center for Numerical Analysis, Univ of Texas at Austin, 1980

3 李晓梅，蒋增荣. 并行算法. 湖南科学技术出版社，1992. 303～307

4 清华大学应用数学系. 现代应用数学手册. 计算方法. 北京：北京出版社，1990. 364～423

The Efficient Implementation of a Library for Large Sparse Linear Algebraic Iterations

He Xinfang　Hu Qingfeng　Wang Liping　Tian Zerong

(Department of Computer Science, NUDT, Changsha 410073)

Abstract

In this paper, we discuss the teration algorithms, acceleration methods, memory techniques and parallel algorithms for large sparse linear equation systems. Combining the features of vector machines, we adopt effective optimization measures and develop the vector library routine.

Key words　speedup, vector, scalar